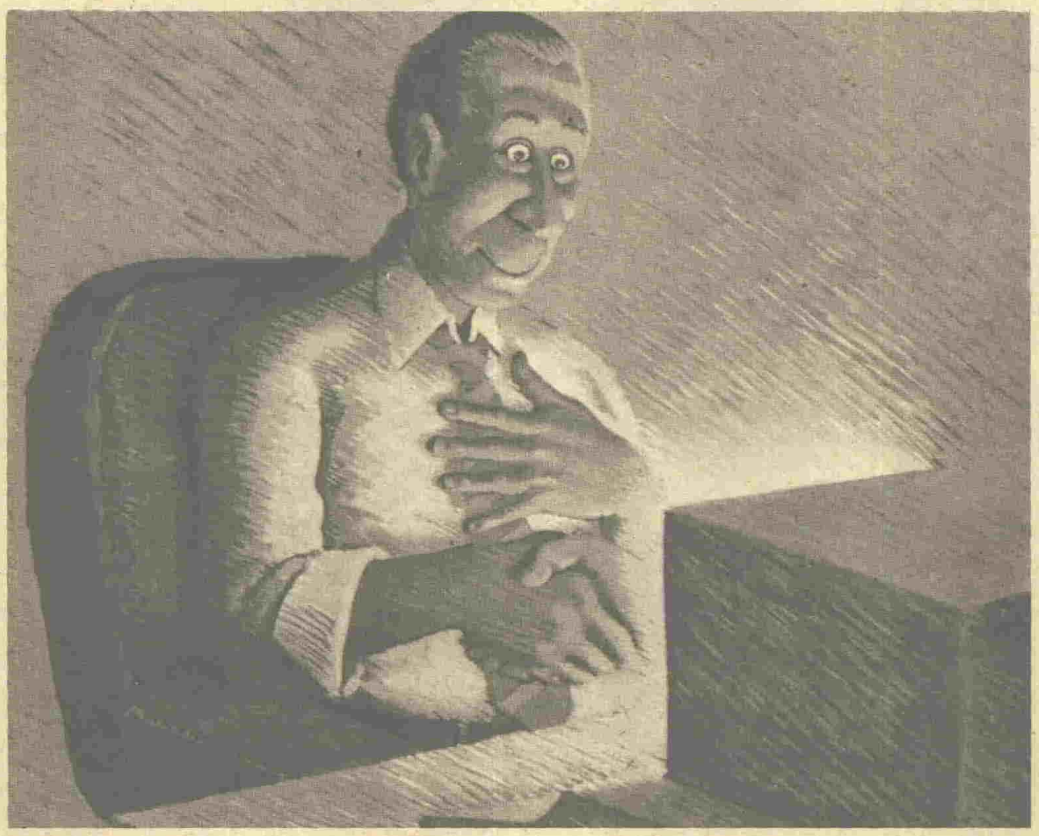
เปิดอก คอมพายเลอร์

คอมพายเลอร์เป็นคำที่รู้จักกันดี แต่มีสักกี่คนที่รู้ถึงการทำงานของคอมพายเลอร์ มาทำความรู้จักกับโครงสร้างของ
- **C** คอมพายเลอร์
- **FORTRAN-H** คอมพายเลอร์
- **BLISS/11** คอมพายเลอร์

กิติพงษ์ ฤทธิบุตร

ในครั้งที่แล้วเราได้พูดกันถึงเรื่องทั่ว ๆ ไปเกี่ยวกับคอมพายเลอร์ ว่าเป็นโปรแกรมชนิดหนึ่งที่ทำหน้าที่หลักในการสื่อสารความต้องการระหว่างผู้ใช้เครื่องคอมพิวเตอร์กับเครื่องคอมพิวเตอร์ ซึ่งจะทำให้เครื่องสามารถเข้าใจความต้องการและดำเนินการตามความต้องการของผู้ใช้ได้ รวมถึงการให้ข่าวสารต่อผู้ใช้ เช่น จำนวนเนื้อที่ที่ใช้ในหน่วยความจำของเครื่องสำหรับโปรแกรม เช่น เนื้อที่สำหรับรหัสข้อมูลที่ใช้และสแต็คเป็นต้น หรือให้ข้อมูลต่อผู้ใช้เกี่ยวกับความผิดพลาดของโปรแกรมที่เขียนขึ้นในกรณีที่มีส่วนใด ๆ ของโปรแกรมคลาดเคลื่อนจากกฎเกณฑ์ของภาษานั้น ๆ หรือข่าวสารอื่น ๆ ที่จะ



เป็นประโยชน์ต่อผู้เขียนโปรแกรมที่จะสามารถแก้ไขโปรแกรมได้ง่ายขึ้น

ในกระบวนการทำงาน (process) ของคอมพายเลอร์ ซึ่งประกอบด้วยขั้นตอน (phases) ที่เริ่มต้นจากการอ่านซอร์สโปรแกรมที่เขียนขึ้นด้วยภาษาคอมพิวเตอร์ภาษาใดภาษาหนึ่ง (เช่น FORTRAN, PASCAL และ C เป็นต้น) ซึ่งจะต้องเป็นไปตามกฎเกณฑ์ของภาษานั้น ๆ จนถึงขั้นตอนสุดท้ายของกระบวนการคือการสร้างรหัสภาษาเครื่อง (object program) ที่มีความหมายเช่นเดียวกันกับซอร์สโปรแกรม

นอกจากนี้ยังได้พูดถึงความจำเป็นหรือประโยชน์ของคอมพายเลอร์ที่มีความสัมพันธ์กับความเจริญก้าวหน้าทางเทคโนโลยีคอมพิวเตอร์อย่างมาก เนื่องจากสามารถอำนวยความสะดวกต่อผู้ใช้ในการติดต่อสื่อสารและสั่งงานกับเครื่องโดยใช้ภาษาระดับสูง ซึ่งเป็นภาษาที่มีความเป็นอิสระต่อเครื่องและมีความสะดวกในการเขียนโปรแกรมมากกว่าการใช้ภาษาระดับต่ำ รวมทั้งมีความเป็นอิสระจากการพัฒนาด้านฮาร์ดแวร์ของระบบคอมพิวเตอร์ด้วย ซึ่งรายละเอียดของเรื่องต่าง ๆ เหล่านี้ท่านผู้อ่านสามารถติดตามได้จากวารสารไมโครคอมพิวเตอร์ฉบับที่ 24 ในเรื่อง **คอมพายเลอร์ ล่ามคอมพิวเตอร์**

ก่อนเข้าเรื่อง

การคุยกันในคราวนี้ก็จะเป็นเรื่องราวของวิธีการออกแบบคอมพายเลอร์ของภาษาระดับสูงบางภาษาที่มีความเด่นเป็นพิเศษ และเป็นเนื้อหาหลักที่เราจะได้กล่าวถึงกันต่อไป แต่ก่อนที่จะไปถึงเรื่องนั้นเราจะพูดกันถึงเทคนิค (translation technique) บางอย่างที่ใช้ในกระบวนการแปลความ (compilation process) ซึ่งวิธีการที่จะนำมาคุยกันนั้นเป็นเทคนิคแบบเรียกใช้ตัวเองตามลำดับขั้น (recursive descent) รวมถึงวิธีการที่นิยมใช้ในการตรวจสอบความถูกต้องของโครงสร้างของภาษาวิธีการหนึ่งที่เรียกว่า parse tree และเหตุผลที่นำเรื่องเหล่านี้มาคุยกันก่อนก็เนื่องจากหัวข้อที่เราจะคุยกันต่อไปนั้น มีความเกี่ยวเนื่องสัมพันธ์กันกับเทคนิคแบบนี้ ดังนั้นถ้าเราได้หยิบยกเรื่องนี้มาพูดคุยกันก่อนก็จะได้ช่วยให้สามารถติดตามเรื่องต่อ ๆ ไปได้ดีขึ้น

สิ่งหนึ่งที่คอมพายเลอร์ทั้งหลายต้องมีคือ เทคนิคการตรวจสอบความถูกต้องของโครงสร้างภาษา ด้วยเทคนิคนี้คอมพายเลอร์จะสามารถฟ้องได้ว่าคำสั่งหรือโปรแกรมที่ใช้อยู่นั้นผิดพลาดที่ใด

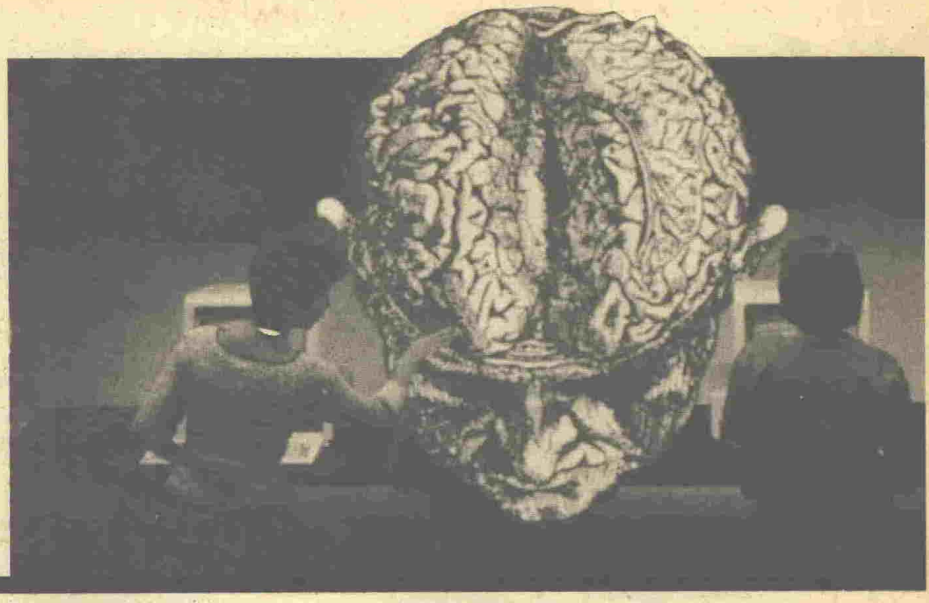
วิธีการเรียกใช้ตัวเองตามลำดับขั้นนั้นมีหลักการพื้นฐานอยู่ที่สิ่งที่จะมีการเรียกใช้ (ในที่นี้ขอเรียกว่า nonterminal) นั้น ต้องมีกระบวนความรองรับเพื่อดำเนินการอย่างใดอย่างหนึ่งตามหลักเกณฑ์ที่ได้กำหนดไว้ เช่น สมมุติว่าเรากำหนดหลักเกณฑ์ไว้ดังนี้

1. <S> ⟶ a<A><S>
2. <S> ⟶ b
3. <A> ⟶ c

<S>,<A>	เป็น nonterminal ที่จะมีการเรียกใช้ตัวเอง
" ⟶ "	เป็นการให้ความหมายหรือการแปลความของ nonterminal
a,b,c	เป็น terminal ซึ่งเป็นผลลัพธ์ส่วนหนึ่งที่ได้จากการปฏิบัติการตามผลลัพธ์ของ nonterminal แต่ละตัว
a<A><S>,b,c	เป็นความหมายหรือผลลัพธ์ของ nonterminal ตามที่ได้กำหนดไว้ในกฎเกณฑ์ของภาษา

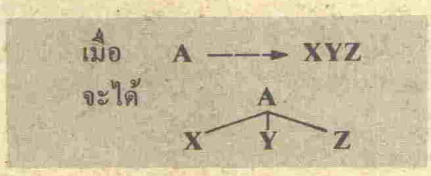
ดังนั้นเมื่อใดก็ตามที่มีการตรวจสอบพบ nonterminal<S>หรือ<A>ก็จะมีการเรียกใช้กระบวนความที่เกี่ยวเนื่องสัมพันธ์กันกับ nonterminal นั้น ในกรณีของตัวอย่างนี้ กระบวนความของ nonterminal <S> จะสามารถเลือกใช้ผลลัพธ์ได้สองแบบ คือ อาจใช้แบบที่ 1 ซึ่งให้ผลลัพธ์เป็น a<A><S>หรือแบบที่ 2 คือได้ผลลัพธ์เป็น terminal b ในการที่กระบวนความ เลือกใช้ผลลัพธ์ใดนั้นก็ต้องขึ้นอยู่กับความเหมาะสมในกระบวนการแปลความ ซึ่งถ้าใช้แบบแรกก็จะต้องมีการเรียกใช้กระบวนความของ nonterminal<A> และ<S>อีก เป็นเช่นนี้เรื่อยไปจนกว่าจะไม่มี nonterminal เหลืออยู่ในขั้นตอนการปฏิบัติงานอีก จะเห็นได้ว่าเทคนิคการเรียกใช้ตัวเองตามลำดับขั้นนั้น ก็คือการให้ความหมายหรือคำจำกัดความของตัวเองแล้วเรียกใช้ตามความเหมาะสม ดังนั้นการออกแบบคอมพายเลอร์ให้สามารถใช้วิธีการแบบนี้ได้นั้น จะต้องให้คำจำกัดความ

การออกแบบโครงสร้างคอมพายเลอร์ ไม่ใช่พิจารณาแต่ทางเทคนิคเท่านั้น แต่ต้องคำนึงถึงปัจจัยอื่นด้านการลงทุนด้วย



ของกฎเกณฑ์ของภาษาที่กำหนดไว้ให้มีการเรียกใช้ตัวเองได้ และภาษาที่จะนำมาใช้เขียนนั้น จะต้องเป็นภาษาที่สามารถให้ทำการเรียกใช้ตัวเองได้ด้วย เช่น ALGOL 68, PASCAL และ C เป็นต้น

สำหรับเทคนิคการตรวจสอบความถูกต้องของภาษาในลักษณะของ parse tree นั้น เป็นวิธีการกระจายโครงสร้างของชุดคำสั่งจากโปรแกรมต้นฉบับเพื่อตรวจสอบว่าเป็นไปตามหลักเกณฑ์ของภาษาหรือไม่ ซึ่งการกระจายนั้นจะแตกออกเป็นส่วนย่อยลงไปเรื่อย ๆ และมีลักษณะของโครงสร้างการกระจายคล้ายกับการแตกกิ่งก้านสาขาของต้นไม้ ตัวอย่าง เช่นถ้าเราสมมุติให้ nonterminal <A> ต้องกระจายออกเป็น <X><Y><Z> (ซึ่งหมายความว่า <X><Y><Z> เป็นผลลัพธ์ของ <A>) เมื่อเราทำการกระจายในแบบของ parse tree แล้วก็จะได้โครงสร้างที่มีรูปร่างตามที่แสดงในแผนภาพ ซึ่ง <X><Y><Z> จะกลายเป็นส่วนย่อย (subtree) ของโครงสร้างแบบ tree นี้



การกระจายตัวแบบนี้จะเป็นไปตามลำดับจากซ้ายไปขวา และเราอาจเรียกส่วนที่กระจายออกมานี้ได้ว่า yield หรือ frontier ของ tree เพื่อให้เห็นได้ชัดเจนขึ้นเรามาสมมุติกันใหม่ว่า เรามีหลักเกณฑ์ของการกำหนดค่าสำหรับภาษาที่ออกแบบไว้ดังนี้

```
Assignment = Variable '='
             Expression;
Expression = Term
           = '-' Term
           = Expression '+' Term;
           = Expression '—' Term;
Term       = Factor
           = Term '*' Factor
           = Term '/' Factor;
Factor     = Constant
           = Variable
           = '('Expression')';
Variable   = Identifier;
Identifier = 'A'! 'B'!.....! 'Z';
Constant   = '0'! '1'!.....! '9';
```

และเรามีชุดคำสั่งจากโปรแกรมต้นฉบับเป็น

$$A = -A + 5 * B / (B - 1)$$

การใช้วิธีการแบบ parse tree เพื่อตรวจสอบความถูกต้องของคำสั่งว่าเป็นไปตามหลักเกณฑ์ของภาษาหรือไม่นั้น ก็จะได้โครงสร้างของ tree ตามรูปที่ 1 ซึ่งวิธีการแบบนี้นิยมใช้ในการทำงานของคอมพายเลอร์ เพราะเป็นวิธีการที่ไม่ยุ่งยากมีการทำงานเป็นขั้นตอน ทำให้สามารถติดตามตรวจสอบได้ง่าย

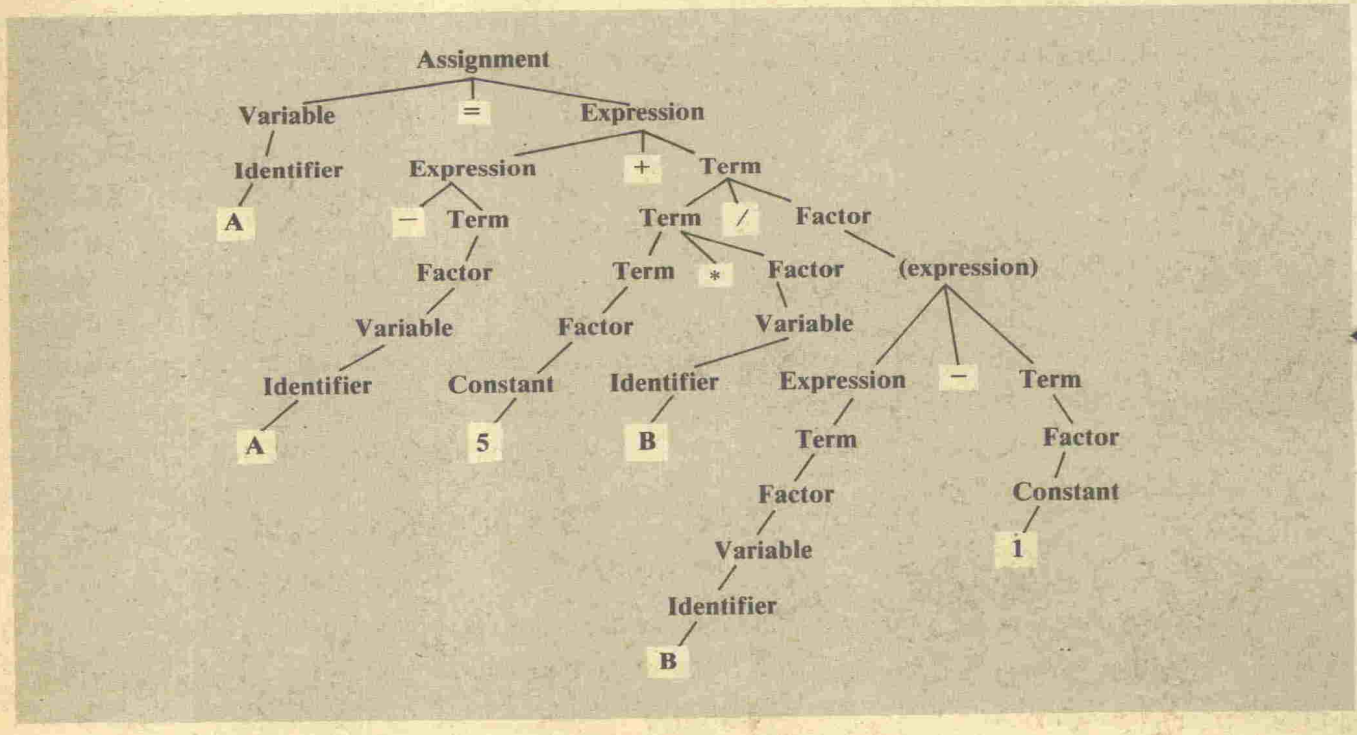
มาดูโครงสร้างของคอมพายเลอร์บางภาษากัน

ภาษาคอมพิวเตอร์ที่เป็นภาษาระดับสูงนั้นมีมากมายหลายภาษา แต่ละภาษาต่างก็มีจุดเด่นหรือวัตถุประสงค์ของการนำไปใช้แตกต่างกันออกไป ตามความต้องการของผู้สร้างภาษานั้น ๆ ซึ่งก็มีผลโดยตรงต่อการกำหนดหรือออกแบบโครงสร้างของคอมพายเลอร์ มีสิ่งต่าง ๆ ที่นอกเหนือจากเทคนิคหรือวิธีการทำงานของคอมพายเลอร์โดยตรงอีกหลายประการ เช่นบุคลากรที่มีอยู่ (ทั้งจำนวนและความรู้ - ความสามารถ) จำนวนงบประมาณและเวลาที่ต้องใช้เป็นต้น ซึ่งปัจจัยต่าง ๆ เหล่านี้มักจะเป็นตัวกำหนดโครงสร้างและขีดความสามารถของคอมพายเลอร์ เนื่องจากการออกแบบคอมพายเลอร์ที่มีโครงสร้างซับซ้อน (เนื่องจากโครงสร้างกฎเกณฑ์ของภาษา) และมีขีดความสามารถมากขึ้นเท่าใด ก็ย่อมหมายถึงการสิ้นเปลืองมากขึ้นเท่านั้น และด้วยเหตุผลเหล่านี้จึงทำให้การสร้างภาษาแต่ละภาษาและคอมพายเลอร์ของภาษาต่าง ๆ เหล่านั้น

จำเป็นต้องเลือกหรือกำหนดให้มีขีดความสามารถเฉพาะทาง หรือมีจุดเด่นในด้านใดด้านหนึ่งมากกว่าที่จะให้ครอบคลุมความต้องการหลาย ๆ ด้าน เพราะการที่จะทำการสร้างภาษาและคอมพายเลอร์ที่สมบูรณ์แบบ *(มีขีดความสามารถสูง สะดวกต่อการนำไปใช้ในงานทุกประเภท ใช้เนื้อที่ในหน่วยความจำของเครื่องน้อย และสามารถสร้างรหัสภาษาเครื่องที่สามารถทำงานได้รวดเร็ว มีประสิทธิภาพสูง)* นั้น จำเป็นต้องใช้ทรัพยากรต่าง ๆ อย่างมาก ซึ่งอาจจะไม่คุ้มค่า

เราลองมาดูตัวอย่างคอมพายเลอร์บางภาษาได้แก่ C compiler, FORTRAN compiler และ BLISS compiler ซึ่งภาษาคอมพิวเตอร์เหล่านี้ต่างก็มีคอมพายเลอร์ที่มีจุดเด่นของตนเองซึ่งแตกต่างกันออกไป ในกรณีของภาษา C นั้นเป็นภาษาที่มีขนาดของคอมพายเลอร์เล็กกว่าเมื่อเทียบกับภาษาอื่น ๆ ในขณะที่สามารถสร้างรหัสภาษาเครื่องได้สั้นและทำงานได้อย่างมีประสิทธิภาพ ส่วน FORTRAN H compiler เป็นคอมพายเลอร์ที่มีจุดเด่นในด้านการปรับปรุงรหัสภาษาเครื่องให้เหมาะสมและสามารถทำงานได้เร็วขึ้นเพื่อไม่ให้มีรหัสที่สั่งการทำงานซ้ำซ้อนกัน และให้ขั้นตอนการทำงานเป็นไปอย่างสอดคล้องกลมกลืน เช่นการลดความซ้ำซ้อนของคำสั่งย่อยที่มีการเรียกใช้บ่อยครั้ง การจัดการเกี่ยวกับขั้นตอนการทำงานในลักษณะของการวนเพื่อทำซ้ำ ตามจำนวนครั้งที่ได้กำหนดไว้ การเปลี่ยนรูปแบบการทำงานของคำสั่งเพื่อให้สามารถทำงานได้เร็วขึ้น เป็นต้น สำหรับ BLISS/11 compiler มีจุดเด่นในด้านการลดขนาดของรหัสภาษาเครื่อง ถือได้ว่าเป็นแบบอย่างที่ชัดเจนสำหรับคอมพายเลอร์ ซึ่งมีวัตถุประสงค์ที่

จุดเด่นของ C คอมพายเลอร์ คือ มีขนาดเล็ก สร้างรหัสภาษาเครื่องได้สั้น และมีประสิทธิภาพ ส่วน FORTRAN H คอมพายเลอร์ ทำงานได้รวดเร็ว และ BLISS/11 คอมพายเลอร์ สามารถลดขนาดของรหัสภาษาเครื่อง ให้ใช้หน่วยความจำน้อยที่สุด



รูปที่ 1 เทคนิคการทำงานแบบ parse tree

ภาษา C เป็นภาษาใหม่ และเริ่มนิยมใช้กันมากขึ้น เป็นภาษาที่ใช้ในการเขียนโปรแกรมจัดระบบงาน UNIX

จะลดขนาดของรหัสภาษาเครื่องให้ใช้เนื้อที่ในหน่วยความจำของเครื่องคอมพิวเตอร์น้อยที่สุด จะเห็นได้ว่าคอมพายเลอร์ทั้งสามภาษานี้ต่างก็มีจุดเด่นที่ต่างกันและอาจจะพูดได้ว่าคอมพายเลอร์เหล่านี้ มีคุณลักษณะที่เด่นกว่าคอมพายเลอร์ของภาษาอื่นๆซึ่งเราจะได้พูดถึงโครงสร้างภายในของคอมพายเลอร์ทั้งสามกันต่อไป

C COMPILER

ภาษา C เป็นภาษาคอมพิวเตอร์ระดับสูงภาษาหนึ่งที่ถูกออกแบบให้สามารถทำงานครอบคลุมลักษณะงานทั่ว ๆ ไป (general purpose programming language) เช่น การคำนวณทางคณิตศาสตร์ การประมวลผลกับข้อมูลที่เป็นข้อความ (strings) การจัดการไฟล์ และการจัดการเกี่ยวกับการนำข้อมูลเข้า (inputs) และการให้ผลลัพธ์ (outputs) ที่ได้จากการประมวลผลเป็นต้น ผู้ออกแบบภาษา C คือ D.M. Ritchie และได้นำไปใช้เป็นครั้งแรกกับระบบคอมพิวเตอร์ที่ใช้ UNIX เป็นโปรแกรมจัดการระบบงาน ซึ่งก็ได้มีการนำภาษานี้ไปใช้กับระบบคอมพิวเตอร์ต่าง ๆ หลายระบบ เช่น PDP-11, Honeywell 6070 และ IBM-370 เป็นต้น

ในปัจจุบันภาษา C เป็นภาษาที่ใช้อย่างแพร่หลายกับระบบคอมพิวเตอร์ทุกขนาดตั้งแต่ขนาดเล็ก เช่น ไมโครคอมพิวเตอร์ขนาด 8 บิท และ 16 บิท ไปจนถึงระบบที่ใหญ่ขึ้น เช่น มินิคอมพิวเตอร์และเมนเฟรม ซึ่งระบบคอมพิวเตอร์ต่าง ๆ เหล่านี้ต่างก็เป็นระบบที่มีโปรแกรมจัดระบบงานแตกต่างกันออกไป โดยที่ภาษา C สามารถใช้ได้กับโปรแกรมจัดระบบงานแทบทุกโปรแกรม การที่ภาษา C ได้รับความนิยมก็เนื่องจากเป็นภาษาที่ได้พัฒนาขึ้นมาเพื่อให้มีความเป็นมาตรฐานมากขึ้น โดยได้รับการออกแบบให้มีความเป็นอิสระจากโปรแกรมระบบงานของระบบคอมพิวเตอร์ต่าง ๆ รวมถึงการออกแบบให้ไม่ขึ้นกับฮาร์ดแวร์อย่างใดอย่างหนึ่งโดยเฉพาะ ซึ่งทำให้สามารถนำไปใช้กับระบบคอมพิวเตอร์ต่าง ๆ ได้เกือบทุกระบบ นอกจากนี้การเขียนโปรแกรมด้วยภาษา C ยังสามารถทำได้อย่างมีขั้นตอน ทำให้สามารถติดตาม ตรวจสอบ แก้ไขโปรแกรมได้ง่าย เพราะเป็นรูปแบบของภาษาที่มีโครงสร้าง ทั้งยังสามารถเขียนโปรแกรมเพื่อใช้แทนภาษาแอสเซมบลีได้ คือโปรแกรมที่เขียนขึ้นสามารถใช้งานได้เหมือนโปรแกรมที่เขียนด้วยภาษาแอสเซมบลี ในขณะที่การเขียนโปรแกรมด้วยภาษา C นั้นทำได้ง่ายและสะดวกกว่า ซึ่งคุณสมบัติต่าง ๆ เหล่านี้ทำให้สามารถนำภาษา C ไปประยุกต์ใช้กับการเขียนโปรแกรมสำหรับงานต่าง ๆ ได้อย่างดี ถึงแม้จะเป็นงานที่มีความสลับซับซ้อนและต้องการความรวดเร็วในการทำงาน เช่น โปรแกรมการจัดระบบของคอมพิวเตอร์ โปรแกรมการจัดการฐานข้อมูล โปรแกรมการประมวลผลข้อความ โปรแกรมจัดการข้อมูลในลักษณะของตารางคำนวณหรือแม้แต่นำมาใช้เขียนโปรแกรมคอมพายเลอร์เอง

ตามที่ได้คุยกันไว้แต่แรกแล้วว่าภาษา C เป็นภาษาที่มีคอมพายเลอร์ขนาดเล็ก และสามารถสร้างรหัสเครื่องจากโปรแกรมต้นฉบับได้สั้น ทำให้สามารถทำงานได้รวดเร็ว ซึ่งหมายความว่าคอมพายเลอร์ของภาษา C นั้น สามารถให้รหัสของโปรแกรมภาษาเครื่องที่มีประสิทธิภาพในการทำงานมากกว่าภาษาระดับสูงอื่น ๆ หลายภาษา ซึ่งเราจะได้มาดูกันว่าโครงสร้างของ C คอมพายเลอร์นั้นมีลักษณะอย่างไร โดยจะได้นำเอาคอมพายเลอร์ของภาษานี้ ที่มีใช้อยู่ในระบบคอมพิวเตอร์แบบ PDP-11 ที่เขียนโดย Ritchie และ Honeywell 6070 กับ IBM-370 ที่เขียนโดย Johnson ว่าได้ถูกออกแบบอย่างไร

การออกแบบ C คอมพายเลอร์เหล่านี้มีลักษณะเหมือนกันที่ถูกกำหนดให้แยกการทำงานเป็นสองส่วน (two-pass compiler) โดยแบ่งกระบวนการทำงาน

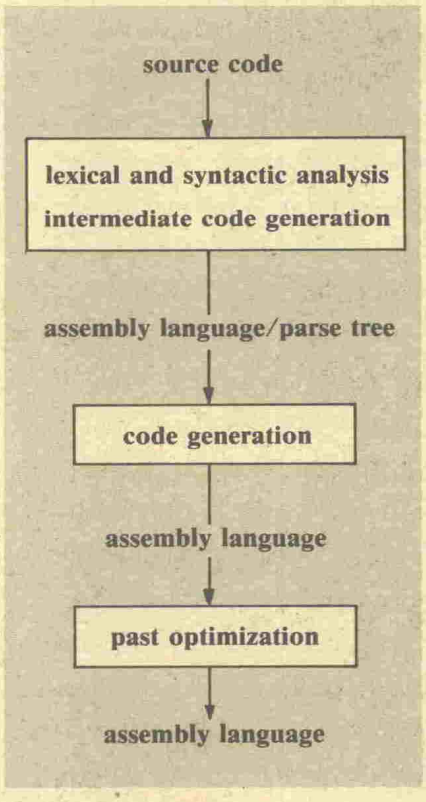
ออกเป็นอิสระจากกัน 2 ส่วน ซึ่งหมายความว่าคอมพายเลอร์จะทำงานในส่วนแรกให้เสร็จก่อนโดยจะจัดการกับโปรแกรมต้นฉบับให้เสร็จทั้งหมดก่อนแล้วจึงเริ่มทำงานในช่วงที่สองกับผลลัพธ์ (intermediate text) ที่ได้จากการทำงานในช่วงแรกต่อไป สำหรับ C คอมพายเลอร์ของระบบ PDP-11 นั้น ถูกออกแบบเป็นพิเศษให้มีส่วนที่ผู้ใช้สามารถเลือกใช้เพิ่มเติม (optional) ได้อีกส่วนหนึ่งด้วย โดยที่ส่วนนี้ (third pass) จะนำไปใช้ในคอมพายเลอร์ด้วยหรือไม่ก็ได้ ซึ่งในส่วนที่เพิ่มขึ้นนี้จะทำหน้าที่ปรับปรุงรหัสภาษาแอสเซมบลีที่เป็นผลลัพธ์จากการทำงานทั้งสองส่วนแรกนั้นให้สามารถทำงานได้ optimization สำหรับโครงสร้างของ C คอมพายเลอร์นั้น ได้แสดงไว้ในรูปที่ 2

การทำงานของ C คอมพายเลอร์เหล่านี้ในส่วนแรก (first pass) นั้นจะประกอบด้วยขั้นตอนต่าง ๆ สามขั้นตอน ได้แก่ lexical analysis, syntax analysis และ intermediate code generation (ถ้าหากคุณ ๆ ผู้อ่านชักจะลืม ๆ ว่าขั้นตอนต่าง ๆ เหล่านี้ทำหน้าที่อะไรกันบ้างละก็ ลองกลับไปทบทวนการทำงานของกระบวนการแปลของ "คอมพายเลอร์" ในตอนที่แล้วก็คงจะเข้าใจได้ดีขึ้น) ซึ่งหมายความว่าในส่วนแรกนี้ C คอมพายเลอร์ จะทำการอ่านคำสั่งจากโปรแกรมต้นฉบับ เพื่อตรวจสอบและแบ่งแยกกลุ่มคำ (token) ตามแบบแผนของภาษา เช่น key word, indentifier หรือสัญลักษณ์และเครื่องหมายทางคณิตศาสตร์เป็นต้น โดยทำการตรวจสอบความถูกต้องตามกฎเกณฑ์ที่ได้กำหนดไว้ แล้วทำการสร้างรหัสของชุดคำสั่งเหล่านั้น ให้อยู่ในรูปแบบของภาษาแอสเซมบลีที่จะประมวลผลตามขั้นตอนการทำงานของเครื่องคอมพิวเตอร์ โดยที่การดำเนินการในส่วนนี้ของ C คอมพายเลอร์จะให้ผลลัพธ์เป็นรหัสแบบ intermediate code ของโปรแกรมต้นฉบับ

C คอมพายเลอร์ จะแยกการทำงานเป็น 2 ส่วนอิสระจากกัน ส่วนหนึ่งเป็นการอ่านคำสั่งจากต้นฉบับ และตรวจสอบความถูกต้อง อีกส่วนหนึ่งจะสร้างผลลัพธ์ที่ได้จากการทำงานในช่วงแรก

ทั้งโปรแกรมเพื่อใช้เป็นข้อมูลของคอมพายเลอร์ส่วนที่สอง (second pass) ต่อไป

สำหรับระบบคอมพิวเตอร์แบบ PDP-11 นั้น C คอมพายเลอร์ ถูกออกแบบให้ทำการเรียกใช้ตัวเองแบบ recursive descent ซึ่งมีขั้นตอนการทำงานตามที่

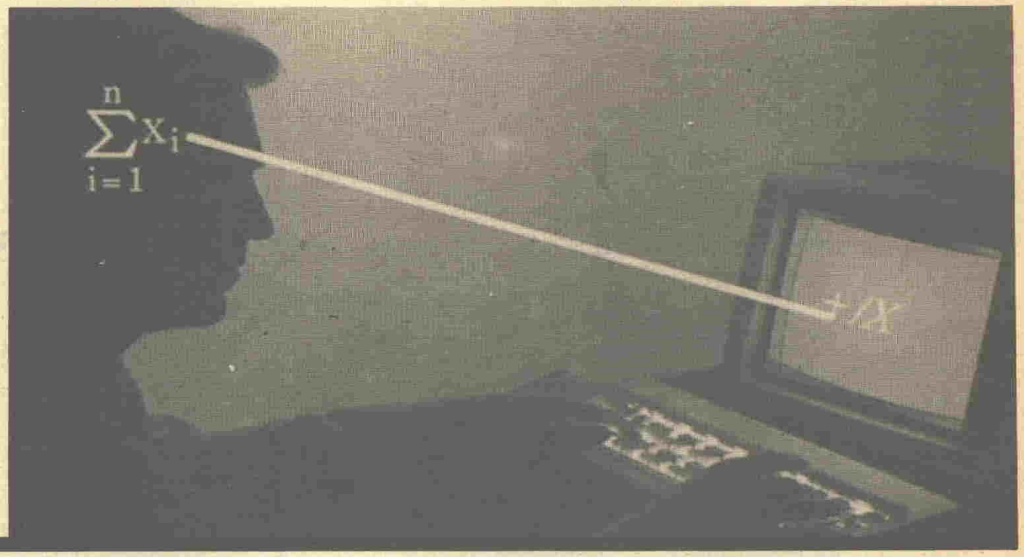


รูปที่ 2 โครงสร้างของ C คอมพายเลอร์

ได้พูดไว้ข้างต้นแล้ว ซึ่งเทคนิคนี้จัดได้ว่าเป็นวิธีการหนึ่งที่จะสามารถทำได้ง่ายและมีขีดความสามารถนำมาใช้ในการเขียนคอมพายเลอร์ได้อย่างดี C คอมพายเลอร์ได้ใช้เทคนิคนี้ในการตรวจสอบความหมายของคำทีละคำ (parse) จากโปรแกรมต้นฉบับ ยกเว้นคำสั่งประเภท expression ซึ่งมีการใช้เครื่องหมายทางคณิตศาสตร์ ที่มีการจัดลำดับความสำคัญก่อนหลัง (operator precedence) เช่น การกำหนดให้เครื่องหมาย "**" (ยกกำลัง) มีลำดับที่สูงกว่าเครื่องหมาย "*" (คูณ) และ "/" (หาร) เป็นต้น

ส่วน C คอมพายเลอร์อีก 2 แบบที่ใช้กับระบบคอมพิวเตอร์ Honeywell 6070 และ IBM-370 นั้น ใช้ขั้นตอนเพื่อการอ่านโปรแกรมต้นฉบับ (lexical analyzer หรือ scanner) ที่ได้จาก LEX ซึ่งจัดว่าเป็นคอมพายเลอร์ที่เป็นเครื่องมืออย่างหนึ่งที่ใช้เพื่อช่วยในการสร้าง lexical analyzer และนับได้ว่าเป็นเครื่องมือที่ช่วยให้การสร้าง lexical analyzer ของคอมพายเลอร์ที่ออกแบบไว้นั้นทำได้ง่ายและสะดวกมากขึ้น ในส่วนของขั้นตอนที่ทำหน้าที่ตรวจสอบความถูกต้องของโครงสร้างภาษา (syntax analyzer หรือ parser) นั้น เป็นส่วนที่ใช้เทคนิคแบบ LALR (1) (lookahead-LR technique) ซึ่งเป็นวิธีใน

C คอมพายเลอร์สำหรับเครื่อง PDP-11 ของ DEC มีส่วนที่ 3 เลือกใช้เพิ่มเติม ให้ปรับปรุงรหัสในส่วนที่ 2 ได้



แบบที่จะทำการอ่านโปรแกรมต้นฉบับจากซ้ายไปขวาและสร้างผลลัพธ์ในรูปแบบที่จะต้องทำการประมวลผลจากขวาไปซ้าย โดย LALR (1) ของคอมพายเลอร์นี้ได้จาก YACC ซึ่งเป็นตัวสร้างพาสเซอร์แบบอัตโนมัติ (automatic parser generator) แบบหนึ่งที่ผู้ใช้สามารถสร้าง parser ได้ด้วยการกำหนดลำดับก่อนหลังของผลลัพธ์ ซึ่งเป็นขั้นตอนที่จะต้องทำการกระจาย nonterminal หรือคำสั่งเพื่อปฏิบัติการอย่างใดอย่างหนึ่งตามหลักเกณฑ์ที่ได้กำหนดไว้ รวมถึงการระบุชนิดของเครื่องที่ต้องการจะนำมาใช้กับคอมพายเลอร์ได้ด้วย

คอมพายเลอร์ของภาษา C ทั้งสามแบบนี้ มี intermediate code ในลักษณะเป็น parse trees สำหรับการตรวจสอบ expressions และชุดคำสั่งที่เป็นรหัสของภาษาแอสเซมบลีตามขั้นตอนการทำงานของโปรแกรมต้นฉบับ (controlflow statements) ในระบบคอมพิวเตอร์ของ IBM และ Honeywell นั้น คอมพายเลอร์จะทำการแทนที่ตัวชี้ตารางสัญลักษณ์ด้วยตำแหน่งของหน่วยความจำที่สัมพันธ์กัน ในขั้นตอนของการสร้าง intermediate code และตารางสัญลักษณ์จะไม่ถูกนำมาใช้อีกในส่วนที่สอง (pass II) ของคอมพายเลอร์

ในขั้นตอนของการสร้างรหัส C คอมพายเลอร์ในระบบของ PDP-11 จะทำ implement ด้วยการใช้วิธีการแบบโครงสร้างแบบต้นไม้ (tree walk) โดยทำการตรวจสอบรีจิสเตอร์ที่จะต้องใช้ในการปฏิบัติงานตามคำสั่งและสำหรับตัวถูกกระทำที่เป็นค่าคงที่ให้เกิดประโยชน์มากที่สุด ส่วนในอีกสองระบบที่เหลือนั้น วิธีการที่ใช้ในส่วนของการสร้างรหัส จะทำการตรวจสอบชุดคำสั่งแต่ละชุดเพียงครั้งเดียว และจะทำการตรวจสอบเพื่อหาวิธีการที่จะดำเนินการกับส่วนย่อยของ tree ให้ได้จำนวนครั้งมากที่สุด โดยที่ไม่ต้องติดต่อกับหน่วยความจำในขณะที่กำลังดำเนินการ แต่จะใช้รีจิสเตอร์ที่ว่างอยู่แทนซึ่งจะมีผลให้การทำงานเป็นไปได้เร็วขึ้น รหัสที่ได้จากการประมวลผลตามวิธีการ subtrees เหล่านี้ จะถูกสร้างขึ้นโดยคอมพายเลอร์ตามรูปแบบของ subtrees ที่ใช้

สำหรับขั้นตอนของการปรับปรุงรหัส (optimization phase) จะทำการสร้างรูปแบบในการแปลเพื่อกำจัดรหัสของชุดคำสั่งที่มีอยู่มากเกินความจำเป็น โดยการจัดขั้นตอนการทำงานให้เหมาะสม เปลี่ยนแปลงคำสั่งที่มีลักษณะกระโดดจากจุดหนึ่งไปยังจุดอื่นที่ให้มีการกระโดดไปทำงานอื่นอีกช่วงหนึ่งหรือการตัดชุดคำสั่งที่ไม่มีการอ้างถึงในขณะประมวลผล เป็นต้น

จนถึงตรงนี้เราก็ได้เห็น C คอมพายเลอร์ที่ใช้ในระบบคอมพิวเตอร์แบบต่าง ๆ กันแล้วว่า เป็นคอมพายเลอร์ที่มีขนาดเล็ก เพราะถูกออกแบบให้มีกระบวนการทำงานเพียงสองส่วน (two-pass) และตัดขั้นตอนบางขั้นตอนออกไป เช่น ในส่วนของ code optimization จึงทำให้มีความจำเป็นที่จะต้องใช้หน่วยความจำของเครื่องไม่มากนัก การสร้างรหัสของ code generator ก็อยู่ในรูปแบบที่ให้มีการใช้รีจิสเตอร์ในขณะประมวลผลมากกว่าที่จะใช้หน่วยความจำ ซึ่งก็ช่วยประหยัดเนื้อที่และเพิ่มความเร็วให้กับการทำงานของออบเจ็กโค้ด และสำหรับ C คอมพายเลอร์ของระบบคอมพิวเตอร์แบบ PDP-11 นั้น มีส่วนที่สาม (third-pass) ให้เลือกใช้เพิ่มเติมได้อีกเพื่อการทำหน้าที่ในการปรับปรุงรหัสที่ได้ในส่วนที่สอง ให้มีขนาดเล็กลงและมีประสิทธิภาพในการทำงานเพิ่มขึ้นอีกส่วนหนึ่งด้วย

FORTRAN H COMPILER

เราได้คุยกันถึง C คอมพายเลอร์มาแล้ว ทีนี้ลองมาดู FORTRAN H คอมพายเลอร์กันบ้าง สำหรับคอมพายเลอร์ที่จะพูดถึงนี้ เป็น FORTRAN คอมพาย-

FORTRAN H คอมพายเลอร์ ใช้กับเครื่อง IBM 360 นับว่าเก่าแก่ที่เดียว เหมาะกับงานทางวิทยาศาสตร์ วิศวกรรม และงานคำนวณ ทางคณิตศาสตร์

เลอร์ที่ใช้กับระบบคอมพิวเตอร์รุ่น IBM 360 ซึ่งมีคุณสมบัติพิเศษที่มีขีดความสามารถในการปรับปรุงรหัสที่แปลจากโปรแกรมต้นฉบับ ให้เป็นโปรแกรมภาษาเครื่องที่สามารถทำงานได้อย่างมีประสิทธิภาพ และอาจกล่าวได้ว่าเป็นคอมพายเลอร์สำหรับการปรับปรุงรหัสที่เด่นตัวหนึ่งทีเดียว ซึ่งเราจะได้คุยกันถึงการทำงานของคอมพายเลอร์ตัวนี้ในส่วนของการทำงานเพื่อปรับปรุงรหัสกันต่อไป

ภาษา FORTRAN เป็นภาษาเก่าแก่ที่ได้รับความนิยมมากภาษาหนึ่งและเป็นภาษาที่ถูกเรียกว่า mathematically หรือ scientifically oriented language เนื่องจากมีความสามารถเป็นพิเศษในงานด้านการคำนวณ นับเป็นภาษาคอมพิวเตอร์ที่นิยมใช้กันอย่างแพร่หลายภาษาหนึ่ง โดยเฉพาะอย่างยิ่งในงานด้านวิทยาศาสตร์และวิศวกรรมศาสตร์ที่มีการคำนวณเป็นหลัก ผู้ออกแบบและนำภาษา FORTRAN มาใช้เป็นครั้งแรกคือบริษัท IBM ซึ่งได้ออกแบบให้สามารถสร้างรหัสภาษาเครื่องที่ใช้ประมวลผลกับข้อมูลตัวเลขได้อย่างรวดเร็วและมีประสิทธิภาพเป็นพิเศษกว่างานด้านอื่น นอกจากนี้ยังง่ายต่อการทำความเข้าใจ โดยเฉพาะอย่างยิ่ง expression ต่าง ๆ ของภาษามีลักษณะคล้ายคลึงกับพีชคณิตมาก ซึ่งทำให้การเรียนรู้การใช้ expression ของภาษามีความเป็นธรรมชาติอย่างมาก

การออกแบบและพัฒนาภาษา FORTRAN ได้เริ่มทำกันมาตั้งแต่กลางปี ค.ศ. 1950 และได้ทำการพัฒนาเรื่อยมา จาก FORTRAN II มาเป็น FORTRAN IV จนในปัจจุบันได้พัฒนามาเป็น FORTRAN 77 ซึ่งได้มีการเพิ่มคำสั่งต่าง ๆ เข้าไปในภาษามากขึ้น ทำให้การเขียนโปรแกรมทำได้สะดวกกว่าเดิม และสามารถนำไปใช้งานเขียนโปรแกรมสำหรับงานในลักษณะต่าง ๆ ได้กว้างขวางยิ่งขึ้น จะเห็นได้ว่าในปัจจุบันระบบคอมพิวเตอร์เกือบทุกระบบไม่ว่าจะเป็นระบบขนาดใหญ่ เช่น เมนเฟรม หรือระบบเล็ก ๆ เช่น ไมโครคอมพิวเตอร์ต่างก็มีคอมพายเลอร์ของภาษา FORTRAN อยู่ด้วย การใช้ FORTRAN คอมพายเลอร์นั้น มักจะใช้งานร่วมกับคำสั่งของ JCL (job control language) เพื่อทำหน้าที่ระบุชนิดของงานที่จะนำไปประมวลผล เช่น การกำหนดอุปกรณ์ร่วม (I/O devices) และระบุตัวผู้ขอใช้ระบบคอมพิวเตอร์เป็นต้น

กระบวนการทำงานของ FORTRAN H คอมพายเลอร์นั้น ถูกออกแบบให้มีการทำงานของโปรแกรมย่อย (sub programs) หลัก ๆ เป็นจำนวน 13 โปรแกรม ซึ่งถูกเรียกว่าเป็น overlays โดยจะทำงานในรูปแบบต่าง ๆ ระหว่างการแปลเพื่อให้ได้ผลลัพธ์ (intermediate text) ตามที่ขั้นตอนการทำงานต่าง ๆ ของคอมพายเลอร์ต้องการ ในจำนวนนี้จะเป็น overlay หลักหนึ่งโปรแกรม ทำหน้าที่เป็นชุดคำสั่งในส่วนสำคัญ ๆ ของกระบวนการแปลร่วมกับขั้นตอนหลักอีกสี่ขั้นตอน ได้แก่ phase 10, 15, 20 และ 25 (ตามรูปที่ 3) ซึ่งแต่ละขั้นตอนนี้เป็นกระบวนการทำงานของแต่ละส่วนของคอมพายเลอร์ด้วย ส่วน overlay ต่าง ๆ ที่เหลืออยู่นั้นจะถูกอ่านเข้าไปเก็บไว้ในหน่วยความจำหลักของเครื่องก็ต่อเมื่อ overlay นั้น มีส่วนเกี่ยวเนื่องสัมพันธ์ หรือต้องใช้ทำงานร่วมกันกับขั้นตอนที่กำลังดำเนินการอยู่ในขณะนั้น ยกเว้นในส่วนของการจัดการเกี่ยวกับความผิดพลาดที่เกิดขึ้นในระหว่างกระบวนการแปล ที่อาจจะถูกอ่านเข้าสู่หน่วยความจำหลักในทุกขั้นตอน ถ้ามีการตรวจพบว่าเกิดความผิดพลาดหรือคลาดเคลื่อนขึ้นในระหว่างการดำเนินการของคอมพายเลอร์

ขั้นตอนการทำงานของ phase 10 นั้น จะทำหน้าที่แทน lexical analysis เพื่อทำการสร้างชุด (list) ของเครื่องหมายทางคณิตศาสตร์และตัวที่จะถูกกระทำ (operator-operand) เป็นคู่ ๆ ในกรณีที่ตรวจพบว่าเครื่องหมายที่ทำหน้าที่แยกคำ เช่น เครื่องหมาย '(' และ ')' เป็นต้น เพื่อให้เห็นได้ชัดเจนขึ้นเราลองมาดูตัวอย่างกันก่อน สมมุติว่าเรามีประโยคคำสั่งดังนี้

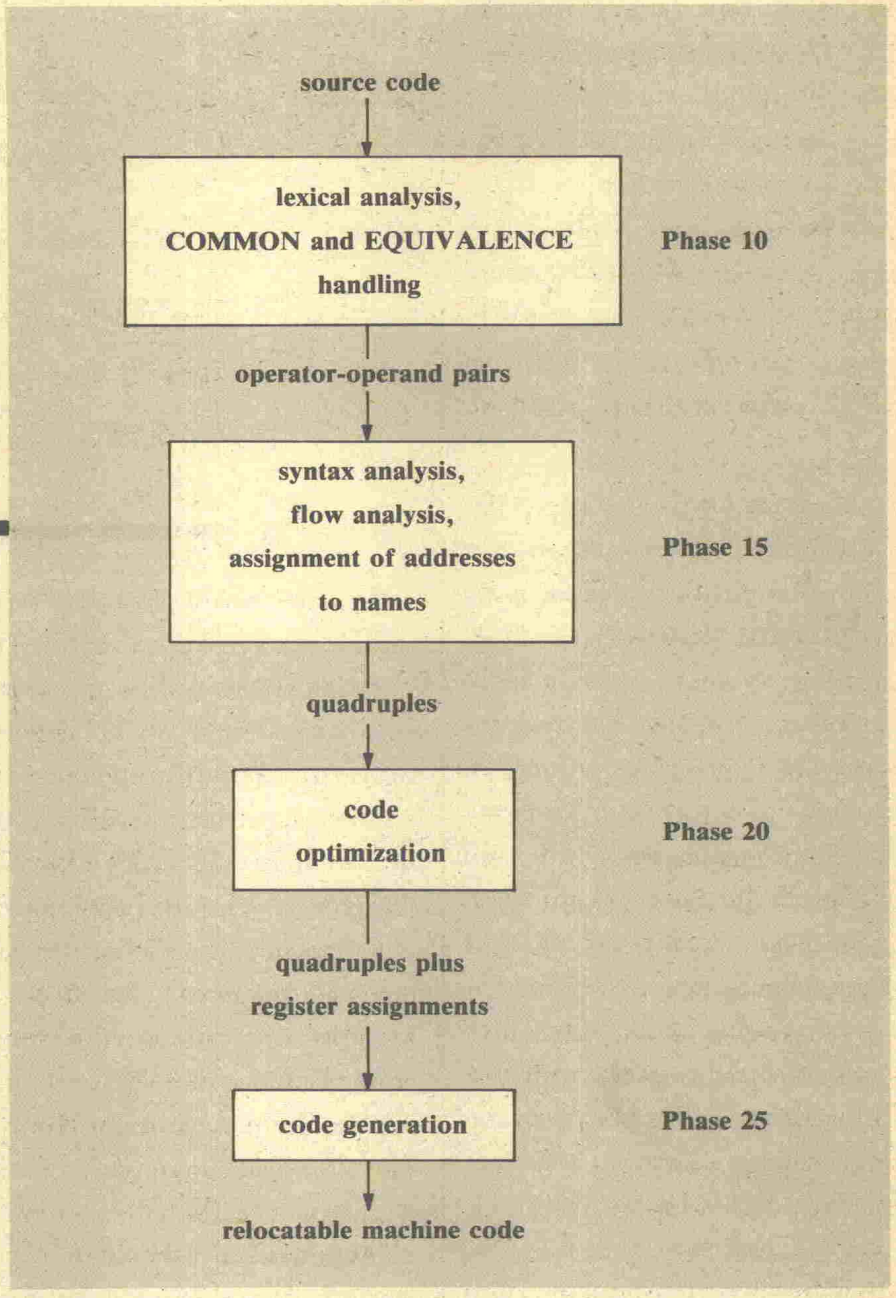
A = B(I) + C

เมื่อคอมพายเลอร์ตรวจสอบแล้วก็จะทำการแปลให้เป็นคู่ ๆ คือ

"assignment statement"	A
=	B
(s	I
)	–
+	C

โดยทั่วไปแล้วในประโยคคำสั่งของ

FORTRAN H คอมพายเลอร์ มีขั้นตอนหลักอยู่ 4 ขั้นตอน และมีโปรแกรม OVERLAY หลัก สำหรับชุดคำสั่งอยู่ 1 โปรแกรม



รูปที่ 3 โครงสร้างของ FORTRAN H compiler

โปรแกรมต้นฉบับจะประกอบด้วยตัว operand แล้วตามด้วยตัว operator สลับกันไปตามลำดับ ซึ่งสามารถใช้เป็นเกณฑ์ขั้นต้นในการพิจารณาความถูกต้องของประโยคคำสั่งในลักษณะนี้ได้ จากตัวอย่างข้างต้นจะเห็นได้ว่ามีลักษณะพิเศษประการหนึ่งคือประโยคคำสั่งมีเครื่องหมายวงเล็บ ซึ่งเป็นเครื่องหมายที่สามารถใช้แสดงความแตกต่างของข้อความในวงเล็บได้ การทำงานในขั้นตอนของ phase 10 ก็จะทำการตรวจสอบเครื่องหมายวงเล็บเปิดว่าเป็นวงเล็บเปิดเพื่อใช้ในความหมายใด เช่นเป็นการใช้ในความหมายแสดงลำดับการคำนวณ หรือเป็นการใช้ในลักษณะของการระบุดัชนี (subscripts) ของตัวแปรที่อยู่ก่อนหน้าเครื่องหมาย และในกรณีนี้คอมพายเลอร์จะให้สัญลักษณ์ของเครื่องหมายวงเล็บเปิดเป็น '(s' เพื่อแทนเครื่องหมาย '(' ในกรณีที่ใช้ในความหมายของดัชนี สำหรับเครื่องหมายวงเล็บปิดนั้นไม่มีความจำเป็นต้องแยกประเภท เนื่องจากทำหน้าที่บอกการสิ้นสุดของประโยคในวงเล็บเพียงอย่างเดียว นอกจากนี้ phase 10 (ในกรณีที่มีการใช้คำสั่งเหล่านี้) ยังทำหน้าที่ตรวจสอบการทำงานของคำสั่ง COMMON และ EQUIVALENCE ซึ่งเป็นคำสั่งที่ใช้จัดการเกี่ยวกับเนื้อที่สำหรับข้อมูลที่จะมีการใช้ร่วมกัน ขั้นตอนหลักที่ทำงานต่อจาก phase 10 ได้แก่ phase 15 ซึ่งจะทำการตรวจสอบความถูกต้องของคำสั่งที่เป็น expressions ว่าเป็นไปตามกฎเกณฑ์ของภาษาตามที่ออกแบบไว้หรือไม่ แล้วทำการเปลี่ยนให้อยู่ในรูปแบบที่เหมาะสมตามวิธีการทำงานของคอมพายเลอร์ต่อไป โดยจะทำการแบ่งแยก expression ต่าง ๆ นั้นออกเป็น 4 กลุ่ม ซึ่งประกอบด้วย

- terminals ซึ่งเป็นกลุ่มของสัญลักษณ์ ที่เป็นผลลัพธ์จากการกระจายของ nonterminal ตามกระบวนการของการแปลความ
- nonterminals ซึ่งเป็นกลุ่มของสัญลักษณ์ที่จะต้องทำการกระจายต่อไป เพื่อให้ได้ผลลัพธ์ใน

ขั้นสุดท้ายตามที่กำหนดไว้ใน production ของ nonterminals นั้น ๆ

- สัญลักษณ์ของจุดเริ่มต้นของกระบวนการแปล
- productions ซึ่งเป็นกระบวนการกระจายตัวหรือการแปลความของ nonterminal เพื่อให้ได้ผลลัพธ์ซึ่งอาจเป็น terminal หรือ nonterminal ต่าง ๆ อีกก็ได้

เนื่องจากภาษา FORTRAN เป็นภาษาที่ไม่มีคำสั่งในลักษณะของโครงสร้าง (structure control statements) เช่น DO-WHILE หรือ IF-THEN-ELSE ดังนั้นการตรวจสอบจึงทำได้ง่าย ยกเว้นในกรณีของ expressions เนื่องจากจะต้องทำการตรวจสอบลำดับความสำคัญก่อนหลังของเครื่องหมายทางคณิตศาสตร์ (operator-precedence) ต่าง ๆ ตามที่ได้กำหนดไว้ในหลักเกณฑ์ของภาษา นอกจากนี้ phase 15 ยังทำหน้าที่ในการแบ่งแยกรหัสออกเป็นกลุ่ม ๆ และทำการสร้างโครงร่างของขั้นตอนการทำงานของโปรแกรมให้เป็นไปตามที่ได้กำหนดไว้ในโปรแกรมต้นฉบับ โดยจะทำการตรวจสอบกลุ่มแต่ละกลุ่มที่ได้แยกไว้ และพิจารณาชื่อต่าง ๆ ที่มีการกำหนดไว้ในโปรแกรม ในขั้นตอนสุดท้ายของ phase นี้ ก็จะเป็นการกำหนดตำแหน่งของหน่วยความจำที่สอดคล้องกัน (relative addresses) ให้กับชื่อต่าง ๆ เหล่านั้น

หลังจากที่ phase 15 ทำการสร้างโปรแกรมที่เป็นรหัสแบบ intermediate code แล้ว ก็จะส่งผลลัพธ์ที่ได้ให้กับ phase 20 ซึ่งรับหน้าที่ในการปรับปรุงรหัส เพื่อทำการเปลี่ยนแปลงแก้ไขรหัสต่าง ๆ ให้อยู่ในรูปแบบที่เหมาะสมและสามารถประมวลผลได้ประสิทธิภาพมากยิ่งขึ้น ในขั้นตอนนี้จะมีการตรวจสอบ expression ย่อยที่จะถูกเรียกใช้ร่วมกันจากส่วนต่าง ๆ ของโปรแกรม และทำการเคลื่อนย้ายหรือเปลี่ยนรูปแบบให้การประมวลผลโปรแกรมที่ได้จากการแปลโปรแกรมต้นฉบับนั้น เป็นไปด้วยความเหมาะสมโดยใช้เนื้อที่ในหน่วยความจำหลักน้อยลงและสามารถทำงานได้เร็วขึ้น ซึ่งการเคลื่อนย้ายนี้จะทำเฉพาะในกรณีที่ตรวจพบว่ามีการกำหนดขึ้นในส่วนย่อยส่วนหนึ่งของโปรแกรม หรือมีการประมวล expression ในส่วนต่าง ๆ เหล่านั้น นอกจากนี้ยังทำการตรวจสอบการทำงานแบบวงรอบ ซึ่งเป็นการทำงานในลักษณะที่มีการทำซ้ำเป็นจำนวนวงรอบตามที่กำหนดไว้ ทั้งในลักษณะขั้นตอนการทำงานและตัวแปรต่าง ๆ ที่เรียกใช้ในวงรอบเพื่อเปลี่ยนแปลงหรือกำจัดส่วนที่ไม่จำเป็นออกไป ในส่วนของ phase 20 นี้จะมีการกำหนดรีจิสเตอร์ต่าง ๆ ให้กับชื่อที่มีการเรียกใช้บ่อยครั้ง ในการทำงานแบบวงรอบสำหรับการเรียกใช้ชุดทำงานย่อยต่าง ๆ นั้น คอมพายเลอร์จะนำรหัสคำสั่งของชุดทำงานย่อยที่ถูกเรียกใช้แทนที่คำสั่งเรียกใช้นั้น เช่น การยกกำลังด้วยเลขจำนวนเต็มก็จะถูกแทนที่ด้วยชุดคำสั่งของการคูณตัวเอง การคูณ/การหาร ก็จะถูกแทนด้วยรหัสคำสั่งที่ให้มีการเลื่อนบิทเป็นต้น

เมื่อผ่านขั้นตอนของ phase 20 แล้ว คอมพายเลอร์ก็จะทำการสร้างรหัสภาษาเครื่อง และส่วนที่ทำหน้าที่เป็น code generator นี้ได้แก่ phase 25 ซึ่งเป็นขั้นตอนสุดท้ายของการทำงานของ FORTRAN H คอมพายเลอร์ รหัสที่ได้หลังจากเสร็จสิ้นการทำงานของขั้นตอนนี้ก็คือโปรแกรมภาษาเครื่องที่สามารถนำไปประมวลผลได้ คือ เป็นรหัสภาษาเครื่องที่มีการกำหนดรีจิสเตอร์ต่าง ๆ ให้กับรหัสเหล่านั้นในระหว่างการประมวลผลเรียบร้อยแล้วนั่นเอง

ขั้นตอนที่เป็นจุดเด่นของ FORTRAN H คอมพายเลอร์ คือ phase 20 ซึ่งเป็นส่วนของเทคนิคที่ใช้ในการปรับปรุงรหัสภาษาเครื่อง

พอถึงตอนนี้เราก็จะเห็นได้ว่า FORTRAN H คอมพายเลอร์ที่ใช้กับระบบคอมพิวเตอร์แบบ IBM 360 นั้น เป็นคอมพายเลอร์ที่ได้ถูกออกแบบให้มีกระบวนการทำงานแบบหลายส่วนซึ่งทำให้คอมพายเลอร์ต้องการใช้เนื้อที่ในหน่วยความจำหลักในการทำงานแต่ละช่วงน้อยลง แต่การทำงานก็จะช้าลงบ้างเนื่องจากการที่มีจำนวน pass มากขึ้น ก็จะต้องใช้เวลาในการอ่าน/เขียนข้อมูลมากขึ้น ในส่วนที่เป็นจุดเด่นของ FORTRAN H คอมพายเลอร์ตามที่ได้พูดถึงในตอนต้นซึ่งได้แก่เทคนิคที่ใช้ในการปรับปรุงรหัสภาษาเครื่องนั้น จะเห็นได้ว่ามี phase 20 เป็นส่วนของ code optimization ที่จะทำหน้าที่นี้ โดยมีลักษณะการทำงานเป็นส่วนหนึ่งของคอมพายเลอร์และจัดเป็นส่วนสำคัญส่วนหนึ่งด้วย

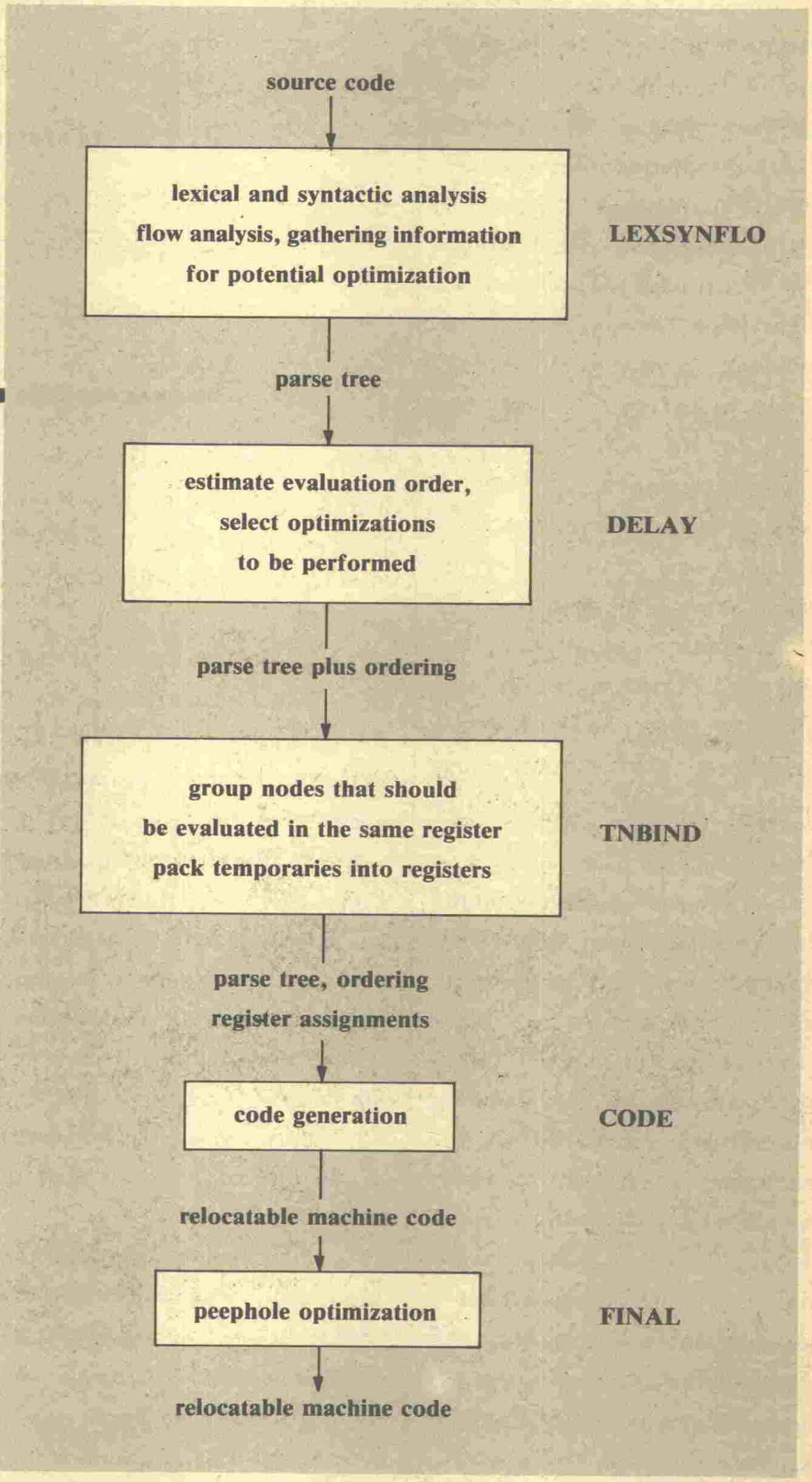
BLISS/11 คอมพายเลอร์ในเครื่อง PDP-11 ที่ประหยัดหน่วยความจำ ประหยัดชุดคำสั่ง และประหยัดเวลา

BLISS/11 COMPILER

คอมพายเลอร์อีกตัวหนึ่งที่เราจะได้คุยกันต่อไปก็คือ BLISS/11 คอมพายเลอร์ที่ใช้กับระบบคอมพิวเตอร์รุ่น PDP-11 ซึ่งมีจุดเด่นอยู่ที่การปรับปรุงรหัสเช่นกัน แต่ไม่ได้มุ่งหมายที่ความเร็วในการประมวลผลของรหัสที่ได้ในขั้นสุดท้ายเหมือนกับ FORTRAN คอมพายเลอร์ เป้าหมายของการเปลี่ยนแปลงรหัสนั้น อยู่ที่การสร้างรหัสที่ต้องการเนื้อที่ในหน่วยความจำน้อยที่สุดเท่าที่จะเป็นได้ อย่างไรก็ตามการลดขนาดที่ต้องการใช้ในหน่วยความจำหลักก็จะทำให้ประหยัดเวลาเพิ่มขึ้นบ้างตามส่วน เนื่องจากจำนวนชุดคำสั่งที่เหลือน้อยลง ย่อมต้องการเวลาในการประมวลผลน้อยลงด้วย

การทำงานของกระบวนการแปลของ BLISS คอมพายเลอร์ มีลักษณะเป็นแบบส่วนเดียว ซึ่งหมายความว่าคอมพายเลอร์จะทำการแปลชุดคำสั่งแต่ละชุดให้เสร็จสิ้นในการอ่านโปรแกรมต้นฉบับเพียงครั้งเดียว แล้วจึงกลับไปอ่านชุดต่อไปใหม่ โดยได้ออกแบบให้มีการทำงานเป็น 5 ส่วน (module) ได้แก่ LEXSYNFLO, DELAY, TNBIND, CODE และ FINAL ตามที่ได้แสดงไว้ในรูปที่ 4

ที่นี้เราลองมาดูกันว่าคอมพายเลอร์



รูปที่ 4 โครงสร้างของ BLISS/11 compiler

ตัวนี้ทำงานอย่างไร สำหรับในขั้นตอนแรกซึ่งเรียกว่า LEXSYNFLO จะทำหน้าที่เป็น lexical analysis และ parsing, การตรวจสอบความถูกต้องของหลักเกณฑ์ภาษานั้นก็ใช้เทคนิคการเรียกใช้ตัวเองตามลำดับขั้นเช่นเดียวกันกับที่ใช้ในคอมพายเลอร์ ภาษานี้เป็นภาษาแบบโครงสร้างที่ไม่มีการใช้คำสั่งประเภท GOTO ดังนั้น การทำงานในลักษณะที่ต้องย้อนกลับไปทำในส่วนที่ผ่านมาแล้วจึงทำในลักษณะของการทำงานแบบวนซ้ำแทน ซึ่งรวมถึงการตรวจสอบ expression ย่อยที่มีการใช้งานร่วมกัน และความสัมพันธ์ของขั้นตอนต่าง ๆ ระหว่างการทำงานภายในวงรอบเพื่อให้การประมวลผลเป็นไปด้วยความเหมาะสมและมีประสิทธิภาพ งานหลักอีกอย่างหนึ่งของขั้นตอนนี้ก็คือตรวจสอบกลุ่มของ expression ที่มีลักษณะคล้ายคลึงกัน โดยจะทำการจัดรูปแบบของ expression เหล่านี้ให้สามารถแทนที่ได้ด้วยชุดคำสั่งเพียงชุดเดียว วิธีการนี้จะช่วยให้ประหยัดเนื้อที่ที่จะต้องใช้ในหน่วยความจำหลัก แต่ก็จะทำให้ต้องใช้เวลาในการประมวลผลเพิ่มขึ้นด้วย

ขั้นตอนต่อไปของคอมพายเลอร์ได้แก่ ขั้นตอนที่เรียกว่า DELAY ในขั้นตอนนี้จะทำหน้าที่ตรวจสอบความถูกต้องของประโยคคำสั่งต่าง ๆ โดยใช้เทคนิคของ parse tree เช่นกัน และจะพิจารณาจาก parse tree ที่ได้ว่ามีส่วนไหนที่สามารถทำการพัฒนาให้ดีขึ้นได้บ้าง และดำเนินการเพื่อขจัดส่วนที่ทำงานซ้ำซ้อนหรือเปลี่ยนแปลงขั้นตอนการทำงานที่ยังไม่เหมาะสม รวมถึงการประเมินลำดับการทำงานของ expression ต่าง ๆ และทำการแก้ไข/ปรับปรุงขั้นตอนให้สามารถใช้รีจิสเตอร์เพื่อเก็บค่าของ expression ย่อยที่มีการใช้งานร่วมกันในระหว่างการประมวลผล สำหรับการตรวจสอบลำดับก่อนหลังของการคำนวณนั้นก็จะใช้กฎเกณฑ์ของพีชคณิตเป็นหลัก

BLISS/11 คอมพายเลอร์ ที่มีขั้นตอนการทำงานค่อนข้างรวดเร็ว ใช้เนื้อที่ในหน่วยความจำน้อย แต่ไม่ได้หมายความว่าโปรแกรมภาษาเครื่องจะสามารถประมวลผลได้รวดเร็วแบบ FORTRAN

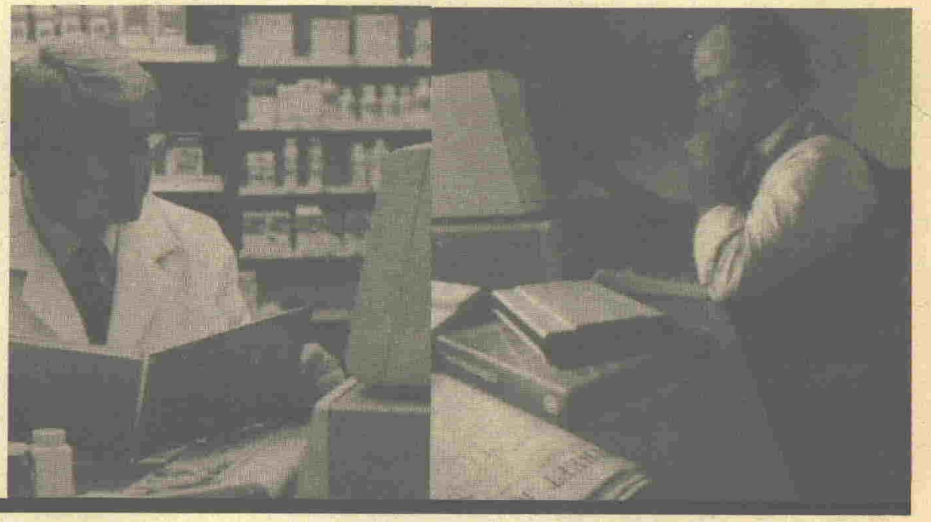
สำหรับ expression แบบมีเงื่อนไขก็จะถูกตรวจสอบทั้งในด้านการคำนวณและขั้นตอนการประมวลผล รวมถึงทำการเลือกวิธีการที่จะสามารถประมวลผลได้ดีกว่าด้วย

ต่อจากขั้นตอนการทำงานของ DELAY แล้ว คอมพายเลอร์ก็จะทำการพิจารณาชื่อต่าง ๆ ที่มีการเรียกใช้ในโปรแกรม เพื่อกำหนดรีจิสเตอร์ให้กับตัวแปรเหล่านั้นว่าชื่อไหนควรจะใช้รีจิสเตอร์ตัวไหนในช่วงใดของการประมวลผล ซึ่งขั้นตอนนี้เรียกว่า TNBIND และวิธีการที่ใช้ในการพิจารณานั้นจะทำโดยการจัดกลุ่มของหน่วยย่อย (nodes) ใน parse tree ที่ควรจะใช้รีจิสเตอร์ร่วมกัน ซึ่งจะทำให้การประมวลผลมีประสิทธิภาพมากขึ้น โดยจะได้ประโยชน์จากการเก็บผลลัพธ์ที่จะใช้ประมวลผลต่อไปไว้ในรีจิสเตอร์ เพราะจะทำให้การเรียกใช้ในครั้งต่อ ๆ ไปทำได้เร็วขึ้น

ผลลัพธ์ที่ได้จากขั้นตอนของ TNBIND จะถูกนำไปสร้างเป็นรหัสตามวิธีการของ code generation ซึ่งเป็นขั้นตอนถัดไปและมีชื่อเรียกว่า CODE ในขั้นตอนนี้จะทำการแปลงโครงสร้างของ tree ให้เป็นรหัสภาษาเครื่องตามลำดับขั้นตอนที่มีการกำหนดรีจิสเตอร์ต่าง ๆ ที่จะต้องใช้กับรหัสเหล่านั้นแล้ว ซึ่งหมายความว่าในขั้นตอนนี้คอมพายเลอร์จะทำการสร้างโปรแกรมภาษาเครื่องที่สามารถจะนำไปทำการประมวลผลต่อไปได้

สำหรับขั้นตอนสุดท้ายจะเป็นการนำโปรแกรมภาษาเครื่องที่ได้จากขั้นตอนก่อนมาทำการปรับปรุงให้มีประสิทธิภาพดีขึ้น โดยจะเป็นการแก้ไขในขั้นสุดท้ายสำหรับการสร้างโปรแกรมที่จะนำไปประมวลผล ซึ่งจะเป็นการกำจัดคำสั่งประเภทที่ให้มีการกระโดดไปทำงานในส่วนที่เป็นคำสั่งให้มีการกระโดดไปทำงานที่จุดอื่นเช่นกัน เพื่อให้การกระโดดไปทำงานที่จุดอื่นของโปรแกรมมีลักษณะที่เป็นไปตามเงื่อนไขของโปรแกรม นอกจากนี้ยังรวมถึงการกำจัดสิ่งที่มีมากเกินไปหรือคำสั่งที่ไม่มีการเรียกใช้ในขณะประมวลผลโปรแกรมนั้น รวมถึงการจัดเรียงลำดับรหัสที่คล้ายคลึงกัน ให้อยู่ในลักษณะของค่าคงที่ที่ใช้เฉพาะส่วนหนึ่งส่วนใดของโปรแกรม ซึ่งจะทำให้การเรียกใช้ทำได้สะดวกขึ้น และจุดสำคัญอีกประการหนึ่งของหน้าที่ในส่วนของ FINAL ก็คือการแทนที่คำสั่งกระโดดซึ่งในระบบคอมพิวเตอร์แบบ PDP-11 ต้องใช้เนื้อที่เป็นจำนวน 2 เวิร์ด โดยใช้คำสั่งเปลี่ยนตำแหน่งแบบ branch instructions ที่ต้องการใช้หน่วยความจำเพียงคำเดียว ซึ่งหมายความว่าจะทำให้ประหยัดเนื้อที่ในหน่วยความจำ

จากจุดเด่นของคอมพายเลอร์แต่ละตัว จะเป็นข้อมูลช่วยในการเลือกภาษาสำหรับการใช้งานเขียนโปรแกรมต่าง ๆ ได้ดีขึ้น



หลักได้อีก

จะเห็นได้ว่า BLISS/11 คอมพายเลอร์ที่ใช้กับระบบคอมพิวเตอร์แบบ PDP-11 นั้น เป็นคอมพายเลอร์ที่ยึดถือการปรับปรุงรหัสเพื่อให้ได้โปรแกรมภาษาเครื่องที่มีความต้องการใช้เนื้อที่ในหน่วยความจำน้อยที่สุดเป็นหลัก โดยที่ในขั้นตอนต่าง ๆ ระหว่างกระบวนการแปลก็จะมีการแก้ไขแทรกอยู่ด้วยเกือบทุกขั้นตอน อย่างไรก็ตามการออกแบบคอมพายเลอร์ตัวนี้ไม่ได้มีวัตถุประสงค์ที่จะสร้างรหัสภาษาเครื่องที่มีความรวดเร็วในการประมวลผล ซึ่งจุดนี้เป็นจุดที่แตกต่างจาก FORTRAN คอมพายเลอร์ที่ยึดถือความเร็วในการทำงานของโปรแกรมภาษาเครื่องเป็นหลัก BLISS/11 คอมพายเลอร์ถูกออกแบบให้มีกระบวนการทำงานช่วงเดียว ซึ่งโดยปกติแล้วจะต้องใช้เนื้อที่ของหน่วยความจำหลักค่อนข้างมาก แต่ก็จะมีผลให้การทำงานของกระบวนการแปลมีความเร็วมากขึ้นตามที่ได้เคยพูดถึงกันมาแล้วแต่ต้น ดังนั้นอาจสรุปได้ว่า BLISS คอมพายเลอร์เป็นคอมพายเลอร์ที่มีขั้นตอนการทำงานค่อนข้างรวดเร็ว และมีความพิเศษในการสร้างรหัสภาษาเครื่องให้มีความต้องการใช้เนื้อที่ในหน่วยความจำน้อย แต่ไม่ได้เน้นในเรื่องของความเร็วในการประมวลผลของโปรแกรมภาษาเครื่อง

ก่อนจบ

จนมาถึงตรงนี้เราก็ได้คุยกันในเรื่องของการออกแบบคอมพายเลอร์สามตัวคือ C คอมพายเลอร์ที่ใช้กับระบบคอมพิวเตอร์แบบ PDP-11, Honeywell 6070 และ IBM 370 อีกตัวหนึ่งก็ได้แก่ FORTRAN H คอมพายเลอร์ที่ใช้กับเครื่อง IBM 360 และสุดท้ายก็คือ BLISS/11 คอมพายเลอร์ที่ใช้กับเครื่อง PDP-11 ซึ่งคอมพายเลอร์ทั้งสามภาษาต่างก็มีจุดเด่นเฉพาะตัว เช่น C คอมพายเลอร์ เป็นคอมพายเลอร์แบบทำงานสองส่วน มีความสามารถในการสร้างรหัส object code ได้สั้นและสามารถทำงานได้อย่างมีประสิทธิภาพ ในขณะที่ FORTRAN H คอมพายเลอร์ เป็นคอมพายเลอร์ที่ถูกออกแบบให้มีการทำงานหลายส่วนและมีความเด่นในด้านการสร้างรหัสภาษาเครื่องที่สามารถทำงานได้อย่างรวดเร็ว (โดยเฉพาะอย่างยิ่งในงานด้านการคำนวณ) ส่วน BLISS/11 คอมพายเลอร์นั้น มีลักษณะเป็นคอมพายเลอร์ที่มีการทำงานเพียงส่วนเดียวและมีความพิเศษในการปรับปรุงเพื่อให้ได้รหัสภาษาเครื่องที่สามารถประหยัดเนื้อที่ในหน่วยความจำหลักในระหว่างการประมวลผลมากที่สุด

ในการออกแบบคอมพายเลอร์นั้น ผู้ออกแบบต่างก็มีจุดประสงค์แตกต่างกันออกไป ทั้งนี้ขึ้นอยู่กับโครงสร้างของภาษาที่ได้กำหนดไว้ก่อนแล้ว รวมถึงทรัพยากรต่าง ๆ ที่ผู้ออกแบบมีอยู่เพื่อใช้ในการสร้างคอมพายเลอร์ เช่น เวลางบประมาณ บุคลากร และความรู้ความสามารถ เป็นต้น อย่างไรก็ตาม การเรียนรู้ถึงวิธีการออกแบบที่ได้เคยมีผู้ทำไว้ก่อนแล้วนั้น ก็เป็นการเรียนรู้ที่ดีวิธีหนึ่งที่จะช่วยให้เกิดแนวความคิดในการสร้างสรรค์ที่อาจจะนำมาประยุกต์ใช้กับการออกแบบคอมพายเลอร์ของเราเอง ซึ่งก็หวังว่าการพบกันในครั้งนี้จะบรรลุความปรารถนาดังกล่าวบ้างตามควร เพราะถึงแม้ว่าจะไม่ได้พูดถึงการออกแบบคอมพายเลอร์เหล่านั้นทุกแง่ทุกมุม และก็คิดว่ามีสาระสำคัญต่าง ๆ ที่พอจะใช้เป็นแนวทางได้

เอกสารอ้างอิง

Alfred V. AHO and Jeffrey D. Ullman **"Principles of Compiler Design,"** Addison-Wesley Publishing Company, 1978.

F.L Bauer and J. Eickel, (Editors), **"Compiler Construction,"** Springer-Verlag Berlin, 1976.

Philip M. Lewis II and Others, **"Compiler Design Theory,"** Addison-wesley Publishing Company, 1976.